VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Fyzikální simulace ve 2D

Physical Simulation on 2D

2010 Lukáš Sojka

Poděkování

Chtěl bych poděkovat svému vedoucímu, Ing. Janu Platošovi, za osobní přístup, odborné vedení mé práce, cenné rady a hlavně za čas, který mi věnoval.

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně.

Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

Datum: 7. 5. 2010 Lukáš Sojka

Abstrakt

Tato bakalářská práce se zabývá fyzikálními simulacemi ve 2D. Fyzikální simulace se v dnešní době stávají užitečnou a nezbytnou součástí v mnoha odvětvích. Tato práce si klade za cíl seznámit se základními fyzikálními jevy a interakcemi, které jsou pro vývoj fyzikálních simulátorů nezbytné. Přiblížit základní problémy při vývoji simulátorů a naznačit exitující řešení. Dále pak navrhnout a implementovat jednoduchý fyzikální simulátor ve 2D v jazyce Java, který bude brát v úvahu popsané jevy. A nakonec provést testy náročnosti tohoto simulátoru na výkon počítače.

Abstract

The Bachelor thesis describes theme physical simulation on 2D. Physical simulation is becoming a useful and necessary part in many departments. This thesis behind aim acquaint with basic physical effects and interaction that are necessary to the development physical simulator. Bring closer basic problems in the development of simulators and suggest existing solutions. Further propose and implement simple physical simulator on 2D in the language Java that takes into account the described physical effects. And the finally, performance tests of this simulator for computer performance.

Klíčová slova

fyzikální simulace, fyzikální engine, počítačové simulace, Java, 2D, fyzika

Key-words

physical simulation, physical engine, computer simulation, Java, 2D, physics

Seznam obsažených obrázků a tabulek

Obsah

1. Úvod

Počítačové simulace se stávají rychle se rozvíjejícím fenoménem dnešní doby. Stávají se užitečnou součástí nejrůznější oblastí a odvětví. Své využití našly již například ve fyzice, chemii, biologii, strojírenství a samozřejmě také v oblasti vývoje počítačových her.

Počítačové simulace se snaží se o vymodelování reálného světa nebo situace tak, aby bylo možné studovat tento „systém“ a vysledovat jak funguje a zkoumat jeho chování. Počítačové fyzikální systémy mohou dnes simulovat reálné fyzikální vlastnosti prostředí, materiálů, různých mechanizmů, elektrických obvodů, tepelných soustav a mnoha dalších. Hlavním důvodem proč se zabývat simulováním reálné fyziky a vyvíjet fyzikální simulátory (enginy) je levné simulování reálných podmínek bez nutnosti provádět testy v reálu. Typickými příklady může být testování konstrukcí automobilů z jak hlediska bezpečnosti tak i z hlediska aerodynamiky. Počítačová simulace je rychlá, lze jí opakovat a poskytuje přesné výsledky, které pak mohou být dále zpracovávány.

Aby toto všechno mohlo být realizováno je potřeba fyzikálních enginů. Fyzikální engine je knihovna tříd a funkcí, která simuluje reálné podmínky a tělesa, které se v nich pohybují, podle Newtonových zákonů. V dnešní době existuje velká řada firem, která se zabývá tvorbou fyzikálních enginů pro konkrétní odvětví. V oblasti herních enginů jsou nejznámější PhysX nebo Havok.

Fyzikální engine je samozřejmě založen na implementaci matematiky a fyziky. Jednotlivé fyzikální enginy (v závislosti na jejich propracovanosti) přistupují k implementaci matiky a fyziky s různou měrou přesnosti.

Základními fyzikální jevy a interakce, které je potřeba znát pro vývoj jednoduchého fyzikálního enginu, jsou popsány v první části mé práce.

V druhé části práce se zabývám analýzou problému fyzikálních simulací. Zde jsou popsány základní problémy a jejich řešení, se kterými je potřeba se vypořádat při tvorbě fyzikálního simulátoru. Dále jsou zde stručně popsány základní dva druhy fyzikálních simulací tj. rigid-body simulace a simulace pomocí tzv. spring-mass systémů.

Třetí část se zabývá popisem řešení některých konkrétních problémů při implementaci mého fyzikálního simulátoru, jako např. detekce kolizí, volba časového kroku. A dále pak obsahuje popis a význam jednotlivých tříd aplikace.

Poslední kapitola se zabývá provedenými testy a jejich výsledky a možnostmi optimalizací vytvořeného simulátoru.

2. Základní fyzikální síly a interakce při simulacích

Fyzika při tvorbě fyzikálních simulacích hraje bezesporu tu nejhlavnější roli. Proto je důležité se na začátku seznámit se základními fyzikálními jevy a interakcemi, které jsou při vývoji potřeba.

Hlavními zdroji informací a definic a fyzikálních vztahů pro tuto kapitolu byly následující zdroje:

1. RNDr. BEDNAŘÍK Milan, CSc., RNDr. ŠIROKÁ Miroslava, CSc., *Fyzika pro gymnázia: Mechanika*. 3. vydání. Praha : Prometheus, 2002. 288 s. ISBN 80-7196-176-0.
2. Doc. RNDr. LEPIL Oldřich, CSc, *Fyzika pro gymnázia: Mechanické kmitání a vlnění*. 3.přepracované vydání. Praha : Prometheus, 2006. 129s. ISBN 80-7196-216-3.
3. *FyzWeb* [online]. Dostupné z WWW: <www.fyzweb.cz>.

2.1 Objekt a jeho vlastnosti (síla, rychlost, poloha,…)

Pro implementaci fyzikálních simulací je potřeba vědět, jak bude vypadat objekt (těleso), který bude simulován. Objekt simulace musí v každém případě mít svou:

- polohu
- rychlost ($\boldsymbol{v}$[1]=[$m \cdot s^{-1}$])
- hmotnost (m=[kg])
- sílu ($\boldsymbol{F}$=[N]), která na objekt působí v daném časovém okamžiku.

podle složitosti fyzikálního simulátoru také poloměr (r), a další.

Jelikož simulace probíhá ve dvourozměrném nebo třírozměrném prostoru a veličiny síly, rychlosti jsou vektorové fyz. veličiny, musí být samozřejmě vektorově vyjádřeny např. $\boldsymbol{v}=(v_x,v_y)$ [1].

Poloha je definována jako bod definovaný svými souřadnicemi ve dvou nebo třírozměrném prostoru (X=[x,y]). Objekt s nenulovou rychlostí se pohybuje ve směru rychlosti. Z tohoto důvodu dochází ke změně polohy v prostoru. Další působící veličinou je čas. Poloha objektu tedy závisí na tom, jak rychle se objekt pohybuje, a na tom kolik času uplynulo od započnutí pohybu. Více o vztazích rychlosti, polohy a času v následující kapitole. O možnostech výpočtu polohy při fyz. simulacích pak v kapitole o problémech fyzikálních simulací.

2.2 Nerovnoměrný pohyb

Mezi základními fyzikálními „jevy“, ve fyzikálních simulacích patří *změny rychlosti tělesa*, *jeho zrychlování*, popř. *zpomalování*. A dále pak také změna polohy tělesa, která je závislá na rychlosti v časovém úseku. Změna polohy tělesa z jedné do druhé se nazývá *dráha*. Tyto změny jsou způsobeny působením okolních sil na těleso. Takovýto pohyb, při něm dochází ke změnám rychlosti tělesa, se nazývá *nerovnoměrný pohyb*.

Nerovnoměrný pohyb je takový pohyb, při němž se mění velikost i směr okamžité rychlosti.

[1] Pozn. Vektorové fyz. veličiny jsou v textu vyznačeny tučně.

Při tomto pohybu dochází ke změně vektoru rychlosti tělesa. Fyzikální veličina, která tuto změnu popisuje, se nazývá *zrychlení* (**a**=[$m \cdot s^{-2}$]). *Zrychlení tedy popisuje změnu rychlosti tělesa v čase.*

Okamžité zrychlení je zrychlení v daném časovém okamžiku. A určuje se jako derivace rychlosti podle času $\boldsymbol{a} = \frac{\boldsymbol{dv}}{dt}$.

Průměrné zrychlení je změna rychlosti za daný časový interval $\boldsymbol{a} = \frac{\Delta \boldsymbol{v}}{\Delta t}$.

V ideální fyzikální simulaci je samozřejmě potřeba znát okamžité zrychlení nebo průměrné zrychlení za co možná nejmenší časový interval Δt. Tohoto však nelze dosáhnout. Více pak v kapitole o problémech fyzikálních simulací.

Z předchozích vzorců pak vyplývá, že *okamžitá rychlost* tělesa je přímo úměrná času a platí $\boldsymbol{v} = \boldsymbol{a}t$.

Rychlost tělesa s počáteční rychlostí $\boldsymbol{v_0}$ je pak $\boldsymbol{v} = \boldsymbol{v_0} + \boldsymbol{a}t$.

Dráha tělesa rovnoměrného pohybu s nulovou počáteční rychlostí $s = \boldsymbol{v}t$.

Dráha tělesa nerovnoměrného pohybu s nulovou počáteční rychlostí je přímo úměrná druhé mocnině času $s = \frac{1}{2}\boldsymbol{a}t^2$ => $s = \frac{1}{2}\boldsymbol{a}t^2$.

Toto byl pohled na nerovnoměrný pohyb z hlediska *kinematiky*. Při fyzikálních simulacích, ale je potřeba vzít také v potaz působení okolních sil na těleso (např. tíhová síla, ...). Tímto se zabývá *dynamika hmotných těles.*

První Newtonův zákon říká: *Každé těleso setrvává v klidu nebo pohybu, pokud není nuceno vnějšími silami tento stav změnit.*

Druhý Newtonův zákon říká: *Jakákoliv změna pohybového stavu je způsobena silovým působením.*

Vztah, který toto vyjadřuje je pak $\boldsymbol{a} = \frac{\boldsymbol{F}}{m}$. Z tohoto pak vyplývá, že rychlost tělesa je ovlivněna silovým působením na těleso.

2.3 Pohyb v tíhovém poli

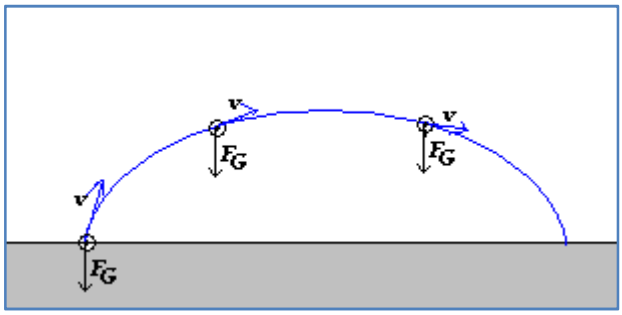
Pohyb v tíhovém poli je další fyzikální „jev“, se kterým je potřeba se při vývoji fyzikálního simulátoru zabývat. Pokud ovšem fyz. simulátor nechce simulovat stav beztíže.

Tíhová síla $\boldsymbol{F_G}$ je síla, která vzniká působením tíhového pole Země na těleso. Její působiště je v těžišti tělesa. Z třetího Newtonova zákona akce a reakce vyplývá, že působení těchto těles je vzájemné. Působení tělesa na Zemi se ovšem zanedbává. Bere v úvahu pouze působení Země na těleso. Tíhová síla Země pak uděluje danému tělesu tíhové zrychlení $\boldsymbol{g}$. Tíhové zrychlení Země je $\boldsymbol{g}$=9.81 m.s-2. Vektorovým zápisem $\boldsymbol{g}$=(0 m.s-2,-9,81 m.s-2).

Pohyby v homogenním tíhovém poli Země jsou:

- *volný pád* – rovnoměrně zrychlený pohyb s nulovou počáteční rychlostí a s konstantním tíhovým zrychlením $\boldsymbol{g}$. Pak zde platí: $\boldsymbol{v} = \boldsymbol{g}t$.
- *vrh tělesa* – těleso koná rovnoměrný přímočarý pohyb ve směru rychlosti a zároveň volný pád ve směru tíhového zrychlení $\boldsymbol{g}$. Složením těchto pohybů vzniká výsledný pohyb, viz obrázek 1. Pak zde platí: $\boldsymbol{v} = \boldsymbol{v_0} + \boldsymbol{g}t$.

Tíhová síla působící na těleso je: $\boldsymbol{F_G = m.g}$.



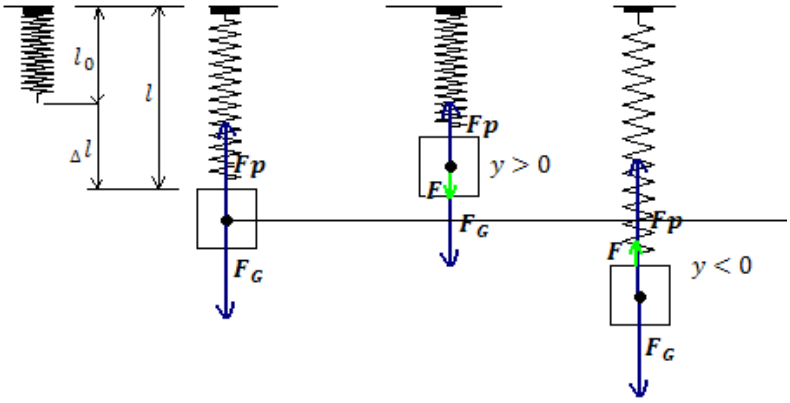
Obrázek 1: pohyb tělesa v tíhovém poli

2.4 Pohyb na pružině

Při některých fyz. simulacích lze využít pružin a pohybu těles na pružinách. Já jsem chování pružin využil při implementaci simulace tělesa na pružině a také při simulaci lana, které je reprezentováno velkou množinou bodů, které jsou mezi sebou pospojovány pružinami.

Těleso zavěšené na pružině tvoří *mechanický oscilátor (pružinový)*. Tento oscilátor *koná kmitavý pohyb*. Kmitavý pohyb je *pohyb nerovnoměrným*, při kterém dochází k neustálé změně rychlosti a zrychlení zavěšeného tělesa.

Tento kmitavý pohyb neustále probíhá okolo *rovnovážné polohy*. Rovnovážná poloha je poloha tuhého tělesa, při níž je výslednice všech působících sil nulová. V případě mechanického oscilátoru na těleso zavěšené na pružině působí dvě stejně velké síly opačného směru. A to *tíhová síla* $\boldsymbol{F_G}$ a *síla pružnosti* $\boldsymbol{Fp}$, viz obrázek 2. V rovnovážné poloze také těleso dosahuje nejvyšší rychlosti. *Síla pružnosti* $\boldsymbol{Fp}$ je tedy síla, která způsobuje kmitání pružinového oscilátoru.



Obrázek 2: kmitání tělesa na pružině

Parametry pružinového oscilátoru, který tvoří těleso zavěšené na pružině, jsou hmotnost tělesa *m* a tuhost pružiny *k*. Kde jednotkou tuhosti je $N \cdot m^{-1}$. Tuhost pružiny vyjadřuje kolik síly $\boldsymbol{Fp}$ je potřeba k prodloužení pružiny o Δl. Vyjádřeno vztahem: $\boldsymbol{k} = \frac{\boldsymbol{Fp}}{\Delta l}$.

Nezatížená pružina má délku l_0. Po zavěšení tělesa o hmotnosti *m* se po jeho ustálení se délka pružiny prodlouží na délku $l = l_0 + \Delta l$.

V rovnovážné poloze musí tedy platit, že výsledná síla $\boldsymbol{F}$, která ovlivňuje změny rychlosti kmitajícího tělesa, je rovna 0, viz obrázek 2. Když se oscilátor uvede do kmitavého pohybu, síla pružnosti $\boldsymbol{Fp}$ *se mění*, zatímco tíhová síla $\boldsymbol{F_G}$ zůstává stále stejná. Výsledná síla $\boldsymbol{F}$, je pak $\boldsymbol{F = Fp + F_G} \Rightarrow \boldsymbol{F = k(\Delta l - y) - mg}$ [2]. Z úprav pak vyplývá nejdůležitější vztah pohybu na pružině: $\boldsymbol{F = -ky}$, kde *y* je výchylka z rovnovážné polohy.

2.5 Tření, pohyb tělesa na nakloněné rovině

Při simulaci tělesa na povrchu je nutno se zabývat silami, které na těleso v danou chvíli působí a ovlivňují ho. Takovými silami jsou *třecí síly* a *síly (nebo spíš rozklad sil), které vychází z naklonění roviny.*

Tření

Tření vzniká, jestliže je těleso v přímém styku s jiným tělesem a pohybuje se tak, že se smýká po povrchu tohoto tělesa. Na styčné ploše obou těles vzniká *třecí síla.* Tato třecí síla vždy *směřuje proti směru rychlosti tělesa*. Rozlišují dva základní druhy tření:

- Smykové tření
- Valivý odpor

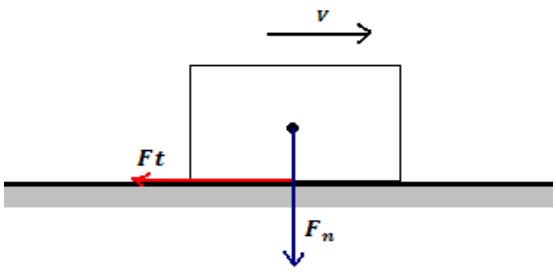
Smykové tření

Je to fyzikální jev, jehož původ je v *nerovnostech stykových ploch*. Při smykovém tření vzniká *třecí síla* $\boldsymbol{Ft}$, jejíž působiště je na stykové ploše obou těles. Třecí síla v reálu nikdy nemůže být zcela nulová.

Velikost *třecí síly* $\boldsymbol{Ft}$ je přímo úměrná velikosti kolmé *tlakové síly* $\boldsymbol{Fn}$. Platí pak vztah: $\boldsymbol{Ft} = f.\boldsymbol{Fn}$, kde *f* je *součinitel smykového tření*.

Je důležité si uvědomit, že *třecí síla nezávisí na obsahu stykových ploch ani na rychlosti* tělesa, které se po povrchu posouvá.

Součinitel smykového tření f je fyzikální veličina, která udává poměr třecí síly $\boldsymbol{Ft}$ a kolmé tlakové síly $\boldsymbol{Fn}$ mezi tělesy při smykovém tření. Závisí na jakosti stykových ploch těles (tj. na materiálech a opracování) a nabývá hodnot 0 – 1. V případě mých fyz. simulací jsem považoval tělesa za ideální tj. *f*=0 a povrch za neideální.



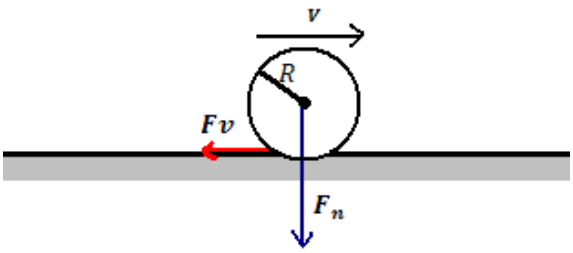
Obrázek 3: smykové tření

Valivé odpor

Je to fyzikální jev, který vzniká, když se *kulové pevné těleso valí po pevné podložce*. Působením tlakové síly ***Fn*** se těleso i podložka mírně deformují. Tato deformace pak vyvolává *odporovou sílu* ***Fv***.

Velikost odporové síly ***Fv*** je přímo úměrná velikosti kolmé tlakové síly ***Fn*** a zároveň na poloměru tělesa R. Platí pak vztah: $\boldsymbol{Fv} = \xi.\frac{\boldsymbol{Fn}}{R}$, kde ξ (ksí) je *rameno valivého odporu*. Čím menší je poloměr tělesa *R*, tím větší je odporová síla.

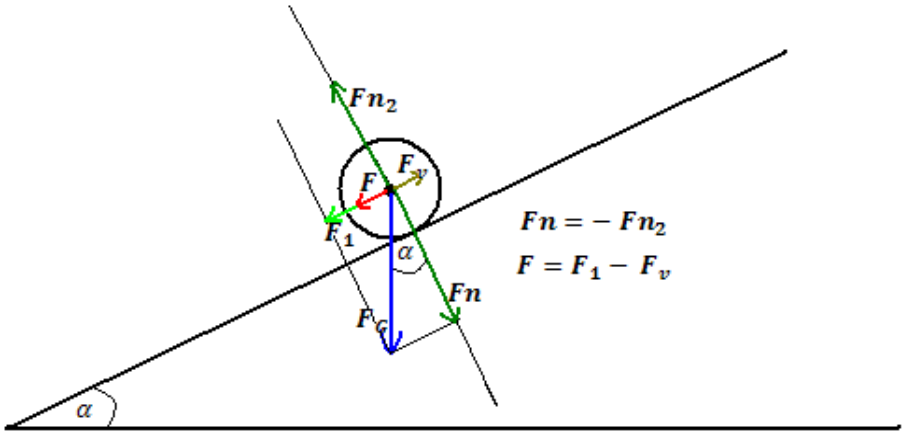
Rameno valivého odporu ξ je fyzikální veličina, která udává poměr velikosti valivého odporu ***Fv*** a kolmé tlakové síly ***Fn*** mezi tělesy při poloměru tělesa *R*. Jeho jednotkou je *metr (m).* Stejně jako součinitel smykového tření závisí na jakostech povrchu a tělesa. Běžné hodnoty ξ se pohybují v tisícinách metrů.



Obrázek 4: valivý odpor

Pohyb tělesa na nakloněné rovině

Až dosud (v kapitole o tření) bylo na síly působící na těleso na povrchu pohlíženo z pohledu vodorovné plochy. Pokud je těleso umístěno na vodorovném povrchu je *svislá tíhová síla* $\boldsymbol{F_G}$ kolmá k povrchu. Z toho vyplývá, že tlaková síla ***Fn***, která ovlivňuje velikost třecí síly ***Ft*** (***Fv***) má stejný směr i velikost jako tato tíhová síla $\boldsymbol{F_{G.}}$ $\boldsymbol{Fn=F_G}$. Při působení *tlakové síly* ***Fn*** na podložku, působí podložka stejně velkou silou opačného směru na těleso $\boldsymbol{Fn_2}$. *Dochází tak k vyrušení působení síly tlakové síly* **Fn** touto silou *a těleso zůstává v klidu.* Pokud ale je těleso umístěno na nakloněnou rovinu, *část tíhové síly se nevyruší s tlakem podložky a dochází ke sklouzávání nebo valení těleso dolu* po nakloněné rovině [3]. Viz obrázek 5.



Obrázek 5: síly působící na těleso na nakloněné rovině

Na těleso na nakloněné rovině působí *tíhová síla* $\boldsymbol{F_G}$, která směřuje ke středu Země. Tuto sílu lze rozdělit do dvou navzájem kolmých sil. A to *síly tlakové* $\boldsymbol{Fn}$ (síla kolmá k povrchu roviny) a *sílu pohybovou* $\boldsymbol{F_1}$(síla rovnoběžná s nakloněnou rovinou), která způsobuje pohyb tělesa dolů po nakloněné rovině, viz obrázek 5.

Tíhovou sílu $\boldsymbol{F_G}$ je určena jako: $\boldsymbol{F_G} = m.\boldsymbol{g}$.

Tlakovou sílu $\boldsymbol{Fn}$ pak jako: $\boldsymbol{Fn} = \boldsymbol{F_G}.\cos\alpha$, kde α je úhel, který svírá nakl. rovina s vodorovnou podložkou.

Pohybovou sílu $\boldsymbol{F_1}$: $\boldsymbol{F_{1=}F_G}.\sin\alpha$ [4].

Pohybová síla $\boldsymbol{F_1}$ *způsobuje pohyb tělesa v ideálním případě.* To je, když nepůsobí žádné třecí síly. Pokud je bráno v úvahu tření, musí být počítáno s třecí silou $\boldsymbol{Ft}$ ($\boldsymbol{Fv}$), která působí proti směru rychlosti. Což může být ve směru síly $\boldsymbol{F_1}$ nebo také naopak[2]. Obecně ale platí, že *výsledná síla* $\boldsymbol{F}$ (určující pohyb tělesa) je: $\boldsymbol{F} = \boldsymbol{F1} + \boldsymbol{Ft}$.

2.6 Srážky těles

Detekce kolizí mezi tělesem a povrchem nebo tělesy navzájem patří k nejsložitějším částem při vytváření fyzikálních simulátorů. Pro správné chování při těchto situacích je důležité, aby se vše dělo podle fyzikálních zákonů. Proto je potřeba zabývat se *srážkami těles*.

Při srážkách těles bude zřejmě záviset na *materiálu těles* a na jejich pohybovém stavu – *hybnosti*.

Hybnost tělesa

Hybnost tělesa $\boldsymbol{p}$=[kg·ms^{-1}] je fyzikální veličina, která vyjadřuje skutečnost, že *pohybový stav tělesa nezáleží pouze na rychlosti, ale také na hmotnosti tělesa*.

Je definována jako součin hmotnosti a okamžité rychlosti tělesa: $\boldsymbol{p} = m.\boldsymbol{v}$. A má stejný směr jako okamžitá rychlost tělesa $\boldsymbol{v}$.

[2] Ve směru síly F_1 by to bylo v případě, že by se těleso válelo směrem vzhůru po nakloněné rovině

Důležitý fyzikální význam má však *změna hybnosti tělesa*. Z předchozích kapitol jsou známy vztahy: $\boldsymbol{F} = \boldsymbol{a}.m$ a $\boldsymbol{a} = \frac{\Delta \boldsymbol{v}}{\Delta t}$. Po dosazení a úpravách pak platí, že $\boldsymbol{F} = \frac{m.\Delta \boldsymbol{v}}{\Delta t}$ => $\boldsymbol{F} = \frac{\Delta \boldsymbol{p}}{\Delta t}$. Z tohoto vyplývá, že změnu hybnosti tělesa lze realizovat silovým působením $\boldsymbol{F}$ po určitou dobu Δt.

Při všech dějích v ideálním reálném světě dochází ke změnám hybnosti tělesa. Tato hybnost se však nikde neztrácí, ale předává se jiných tělesům. Z toho vyplývá, že platí *zákon zachování hybnosti*.

Zákon zachování hybnosti říká, že při všech mechanických dějích *je celková hybnost soustavy stále stejná (konstantní)*. Aritmetický součet všech hybností před dějem je stejný jako součet hybností po ději. Platí pak vztah: $\boldsymbol{p} = \boldsymbol{p_1} + \boldsymbol{p_2} + \cdots + \boldsymbol{p_n}$.

Srážky

Pro pochopení srážek těles je nutné znát ještě další dva pojmy a to *kinetická energie* a *potenciální energie*.

Potenciální energii E_p má každé těleso zvednuté do určité výšky *h*. Platí: $E_p = mgh$.

Kinetickou energii E_k má pohybující se těleso o určité hmotnosti. Platí: $E_k = \frac{1}{2}mv^2$.

Ideálně pružná srážka

Ideálně pružná srážka je srážka, při které se energie *vyměňuje bezztrátově* mezi *potenciální a kinetickou energií*. Jinak řečeno kinetická energie před a po srážce je stejná. Při ideálně pružné srážce se zanedbává úbytek energii spotřebovaný na deformaci těles. *Součinitel restituce (viz níže)* je roven 1.

Např. těleso, které dopadá na zem, se od ní odráží se stejně velkou rychlostí, s jakou dopadlo $\boldsymbol{v} = \boldsymbol{w}$.

Obecná (nedokonale pružná) srážka

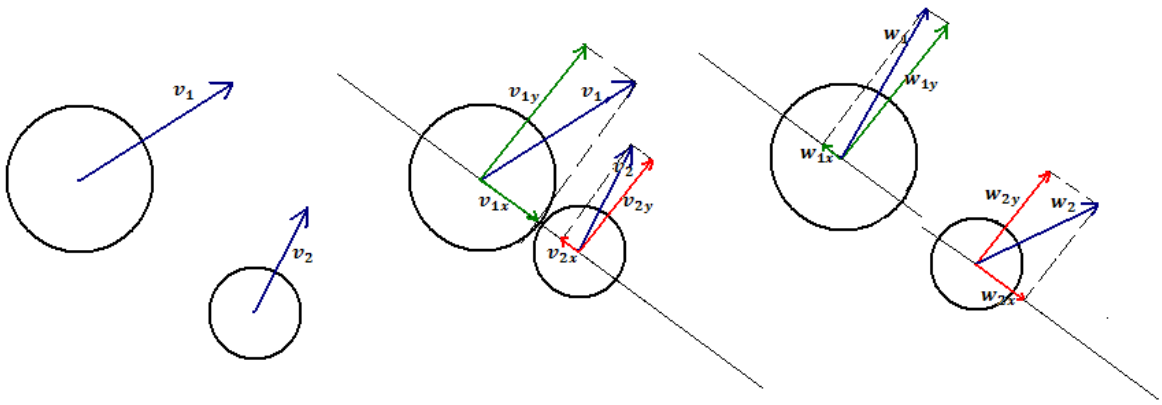
Při takových to srážkách *dochází ke ztrátám energie deformací těles při odrazu*. Pro kvantitativní posouzení pružnosti srážky se zavádí veličina nazývaná *součinitel restituce*.

Součinitel restituce (vzpruživou) k je definován jako poměr relativních rychlostí těles před a po srážce. Hodnota *k* je charakteristická pro každé dva materiály a nabývá hodnot od 0 do 1. – $\boldsymbol{k} = \frac{\boldsymbol{w_1} - \boldsymbol{w_2}}{\boldsymbol{v_1} - \boldsymbol{v_2}}$.

Srážky dvou těles – šikmý ráz

Zde budou popsány srážky dvou těles (koulí) blízké svou hmotností a poloměrem (ne tedy srážka tělesa a Země). A to z *pohledu ideálně pružné srážky* (ideálně nepružná srážka dvou sobě podobných těles je příliš složitá).

Podmínkou takovéto srážky jsou, již výše zmíněné, *zákony zachování energie a hybnosti*. Při srážce se rychlosti těles $\boldsymbol{v_1}$ a $\boldsymbol{v_2}$ rozkládaní do dvou navzájem kolmých směrů $\boldsymbol{v_{1x}}$, $\boldsymbol{v_{1y}}$, $\boldsymbol{v_{2x}}$, $\boldsymbol{v_{2y}}$. Viz obrázek 6.



Obrázek 6: rozlad sil při šikmém rázu

Pro *x-ovou* soustavu budou uplatněny výše zmíněné zákony.

Takže: $\boldsymbol{p_x = p_{1x} + p_{2x}}$ a $\boldsymbol{E_k = E_{k1x} + E_{k2x}}$, po několika úpravách dostaneme:

$$\boldsymbol{w_{1x} = \frac{(m_1 - m_2)v_{1x} + 2m_2v_{2x}}{m_1 + m_2}}$$

$$\boldsymbol{w_{2x} = \frac{(m_2 - m_1)v_{2x} + 2m_1v_{1x}}{m_1 + m_2}}$$

Pro y-ovou soustavu platí:

$$\boldsymbol{w_{1y} = v_{1y}}$$

$$\boldsymbol{w_{2y} = v_{2y}}$$

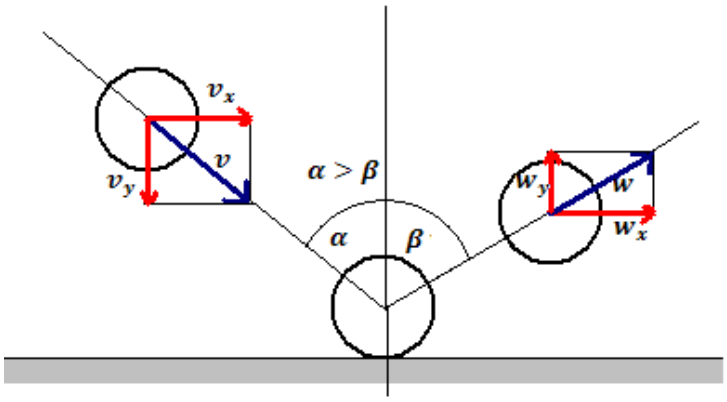
Více informací na [5] a [6].
Viz obrázek 6.

Odraz tělesa od povrchu

V tomto případě jde o *srážku tělesa (koule) s pevným povrchem*. Povrch si lze představit jako druhý předmět o velmi velké hmotnosti.

Při ideálně pružné srážce dochází k rozkladu rychlosti $\boldsymbol{v}$ do dvou navzájem kolmých směrů $\boldsymbol{v_x}$, $\boldsymbol{v_y}$. Při srážce rychlost na ose x zůstává zachována $\boldsymbol{w_x = v_x}$, na ose y má opačný směr $\boldsymbol{w_y = -v_y}$. Z toho vyplývá, že úhel odrazu se rovná úhlu dopadu.

Při nedokonale pružné srážce opět dochází k rozkladu rychlosti $\boldsymbol{v}$ do dvou navzájem kolmých směrů $\boldsymbol{v_x}$, $\boldsymbol{v_y}$. Při srážce rychlost na ose x zůstává zachována $\boldsymbol{w_x = v_x}$, kdežto kolmá složka $\boldsymbol{v_y}$ bude po srážce v důsledku nedokonalé pružnosti zmenšena na hodnotu $-k\boldsymbol{v_y}$, kde k je již zmíněný součinitel restituce. Úhel odrazu je v tomto případě menší než úhel odrazu [6]. Viz obrázek 7.



Obrázek 7: odraz tělesa při nedokonale pružné srážce

2.7 Odpor prostředí

Pro dosáhnutí realističtější fyzikální simulace, by se nemělo zapomínat na odpor prostředí. Prostředím se rozumí plynná nebo kapalná látka.

Při pohybu tělesa prostředím dochází k tzv. *obtékání*. Při tomto obtékání dochází v důsledku vnitřního tření kapaliny nebo plynu k vzniku *odporových sil*, které *působí proti směru pohybu tělesa v prostředí*. U kapalin se mluví o *hydrodynamické odporové síle* u plynů o *aerodynamické oporové síle*.

Pro velikost *aerodynamické odporové síly* platí vztah: $\boldsymbol{F} = \frac{1}{2} C \rho \boldsymbol{v}^2$, kde C je součinitel odporu, ρ je hustota vzduchu, S obsah průřezu tělesa kolmého ke směru pohybu a $\boldsymbol{v}$ rychlost tělesa.

Součinitel odporu C je závislý na tvaru tělesa. Pro kouli je C=0,48.

Hustota vzduchu je přibližně 1,23 $kg \cdot m^{-3}$ [7].

3. Analýza problému fyzikálních simulací

Předchozí kapitola byla zaměřená na fyzikální síly a interakce, které je potřeba implementovat při tvorbě fyzikálního simulátoru. Tato kapitola se bude zabývat možnostmi realizace těchto fyzikálních záležitostí a problémy, které při tom nastávají.

3.1 Fyzika a počítače

Na začátku této kapitoly je dobré připomenout dvě nejdůležitější rovnice pro fyzikální simulace z předchozí kapitoly. Je to rovnice Newtonova pohybového zákona: $\boldsymbol{F} = m.\boldsymbol{a}$ a rovnice pro výpočet *okamžitého zrychlení* $\boldsymbol{a} = \frac{\boldsymbol{dv}}{dt}$. Okamžité zrychlení je tedy rovno derivaci rychlosti podle času. Takovéto zrychlení *vyjadřuje zrychlení v nekonečně malém časovém okamžiku.*

Pro vývoj ideálně přesného fyzikálního simulátoru by bylo potřeba znát toto *okamžité zrychlení* $\boldsymbol{a}$. Pro dosažení takovéto okamžitého zrychlení je potřeba řešit *diferenciální rovnice pohybu*. Zde nastává problém. *Počítače totiž nejsou schopny* v reálném čase efektivně analyticky řešit diferenciální rovnice pohybu.

K simulaci se proto typicky *přistupuje diskrétně*, tj. po krocích [8]. Dochází k tomu, že časový úsek je rozložen do *stejně velkých menších časových úseků dt* (Δt). V tomto časovém úseku pak probíhá *výpočet zrychlení* (již průměrného za časový okamžik *dt*) a dalších operací a výpočtů simulace.

Při takovém to postupu, již dochází ale zcela zřetelně k *nepřesnostem při simulaci*. Viz příklad.

Příklad:
Nějaké těleso (např. auto) se rozjíždí z nulové rychlosti. V čase t_1=5s má rychlost v_1=10m.s^{-1}. Jakou rychlost mělo auto v čase t_2=3s? Zrychlení stanovíme $\boldsymbol{a} = \frac{\boldsymbol{dv}}{dt}$ => $\boldsymbol{a} = 1{,}4\ m.s^{-2}$. Rychlost pak $\boldsymbol{v} = \boldsymbol{a}.dt$ => $\boldsymbol{v} = 4m.s^{-1}$. Tato vypočtená rychlost neodpovídá skutečné velikosti rychlosti v čase 3s, neboť hodnota zrychlení $\boldsymbol{a} = 1{,}4\ m.s^{-2}$ je *střední hodnotou zrychlení* za čas *dt.* Z reality víme, že při rozjezdu automobilu se zrychlení postupně zvětšuje, není okamžitě konstantní [9].

Řešením takovéhoto problému by mohlo být stanovit si časový okamžik *dt, co možná nejmenší*, blížící se nekonečnu. V takovém to malém časovém okamžiku by získané hodnoty byly skoro zcela přesné. Zde se ale naráží na několik dalších problémů plynoucích z *omezenosti dnešních počítačů*, které zabraňují použití tak malého časového kroku. Jsou to:

- *Omezená paměť* – počítače neznají pojem nekonečno a nemohou mít v paměti uloženo nekonečné množství hodnot, nebo i velké množství hodnot blížící se nekonečnu.
- *Časová náročnost* – počítač by musel pro každý takový malý krok provést množství výpočtů. Důsledkem by bylo např. neplynulost simulace.
- *Zaokrouhlovací chyby* – Počítače pracují s čísly s omezeným počtem desetinných míst. Při velmi málem časovém úseku *dt*, blížícímu se nekonečnu, by se změna hodnoty projevovala až na nekonečně vzdáleném desetinném místě. Počítač takové hodnoty zaokrouhluje. Tím by se paradoxně mohlo stát, že při velmi malých intervalech by nepřesnost byla větší než při intervalech větších [10].

Řešením, které se při vývoji fyzikálních simulátorů používá, je přistupovat k simulaci po vhodně velkých krocích, které se pak aproximují pomocí některé z *numerických metod*.

3.2 Numerické metody – Eulerova metoda

Při fyzikálních simulacích se využívají *numerické metody*. Numerické metody slouží k *aproximaci jednotlivých kroků simulace*. Snaží se, co nejblíže přiblížit přesnému výsledku.

Jednotlivé metody se od sebe liší svou *přesností* a svou *náročností na výpočet*. Záleží vždy na konkrétním případu použití. Např. při fyzikálních hrách nám nejde až tak o přesnost, ale spíše o plynulost simulace. Někde zase naopak chceme simulaci s přesnými výsledky.

Důležité pro numerické metody je znát *počáteční stav*. To znamená, kde je těleso umístěno, jakou má rychlost, jaká na něj působí síla. Vychází se vždycky z hodnot, které jsou známé, *a určuje se stav o krok dál* [9].

Nový stav počítají numerické metody vždy po nějakém *kroku*. Tento krok může být proměnlivý nebo taky ne.

Důležité je si uvědomit, že žádná numerická metoda *nebude zcela přesná*.

Numerických metod je mnoho. Nejjednodušší a nejrychlejší je *Eulerova metoda*. Tuto metodu jsem použil také já ve své práci. Mezi další patří metoda *Runge-Kutta, Verletova metoda a další.*

Eulerova metoda

Jak již bylo zmíněno, Eulerova metoda je *nejjednodušší a nejrychlejší a nejméně výpočetně náročná* numerická metoda. Samozřejmě je *také nejméně přesná*. Využívá se většinou v počítačových hrách.

Je vyjádřena toto rovnicí: $x(t_0 + h) = x(t_0) + hx(t_0)$, kde *t* je čas a *h* je časový krok [14].

Stanovení rychlosti a polohy tělesa Eulerovou metodou

Při použití Eulerovy metody při fyzikální simulaci tělesa pak dostáváme rovnice pro stanovení rychlosti tělesa v *novém stavu* (po posunu o časový krok *h* (*dt*)):

$\boldsymbol{v}(t_0 + h) = \boldsymbol{v}(t_0) + h.\boldsymbol{a}(t_0)$, kde $a = \frac{F(\boldsymbol{t_0})}{m}$,

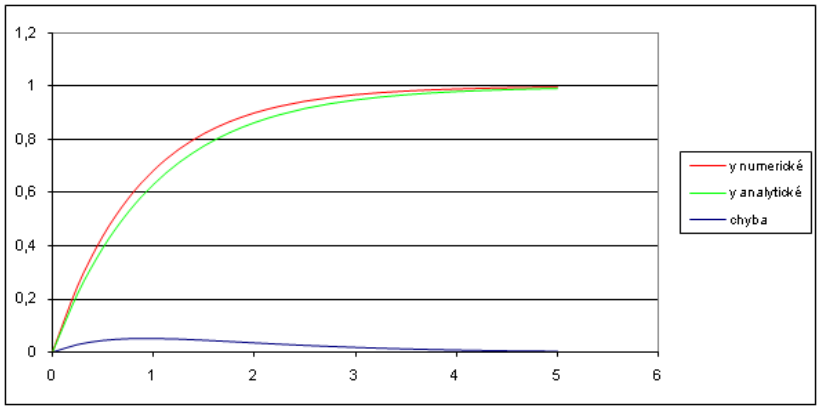
a pro stanovení polohy tělesa v *novém stavu*:

$\boldsymbol{x}(t_0 + h) = \boldsymbol{x}(t_0) + h.\boldsymbol{v}(t_0)$,

čas v *novém stavu* pak je: $t = t_0 + h$.

Nepřesnost Eulerovy metody

Nepřesnost Eulerovy metody roste přibližně s druhou mocninou kroku [9]. Jak je možno vidět na obrázku 8.



Obrázek 8[3]: nepřesnost E. metody při *h*=0,25

3.3 Druhy fyzikálních simulací

V této kapitole by byly popsány dva základní druhy fyzikálních simulací. Prvním a nejznámějším jsou *simulace pomocí tuhých těles (rigid-body simulace)*, založené na tuhých nedeformovatelných tělesech. Druhý přístup jsou simulace založené na *modelování těles jako množině hmotných bodů spojených pružinami (spring-mass systémy).*

Rigid-body simulace

Simulace na základě tuhých těles (rigid-body simulace) je založena na tom, že všechna simulovaná tělesa jsou *tuhá tělesa*. Tyto simulace jsou celkem přesné. Tření a odrazy a jiné výpočty jsou počítány korektně.

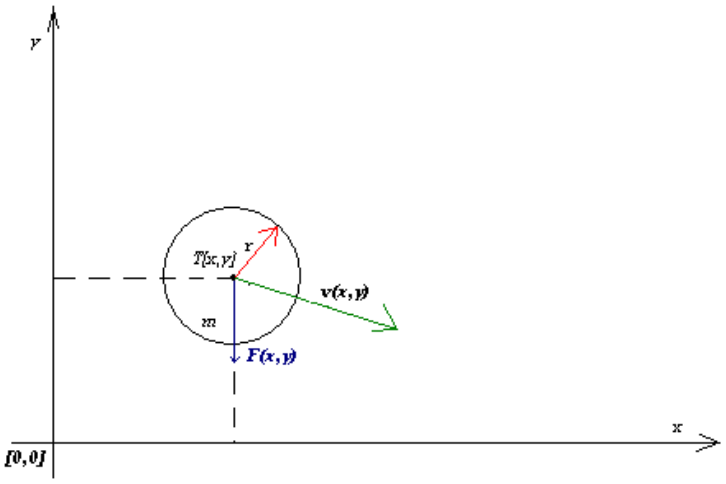
Tuhé těleso je těleso, jehož tvar ani objem *se působením sil nemění*. Tj. nedeformuje se. Všechny síly, které na těleso působí, mají *pouze pohybový účinek*. Hmotnost tuhého tělesa je rovnoměrně rozložena po celém tělese (např. nádoba naplněná do poloviny vodou nemůže být tuhé těleso). Jeho *těžiště je neměnné*. Vzdálenost mezi všemi body (částicemi tělesa je stále stejné). Tuhé těleso ve skutečnosti neexistuje [11].

Z definice tuhého tělesa plynoucí omezení ovšem pro kvalitu fyzikálních simulací nejsou příliš omezující a nijak významně nezhoršují simulaci. To, že se těleso v simulaci viditelně deformuje, je už grafická záležitost.

Tělesa v těchto simulací jsou přesně *určena těžištěm*. Těžiště je dvou nebo tři rozměrný bod. *Srze toto těžiště je s tělesem manipulováno (nastavována rychlost, síla, atd.).* Ostatní parametry jako např. poloměr tvoří „obálku“ kolem těžiště. V případě kolizí tvoří poloměr tělesa „*kolizní obálku*“. Další parametry jako hmotnost, odporové konstanty tělesa jsou skalární veličiny [10]. Na obrázku 9. je ukázka nejjednoduššího rigidního tělesa v prostoru.

Více o problémech rigid-body simulací (detekce kolizí,…) přímo v kapitole o implementaci fyz. simulátoru.

[3] Obrázek převzat z adresy: http://home.zcu.cz/~polreich/NM/NM_euler_RK.pdf



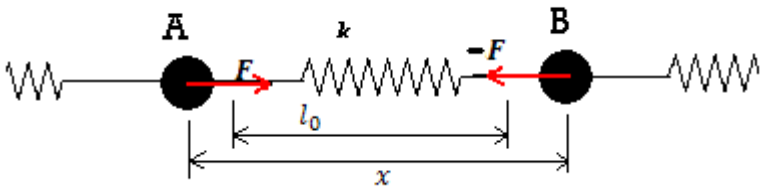
Obrázek 9: rigidní těleso

Spring-mass systémy

V tomto typu fyzikálních simulací se objekty simulace modelují *jako množina bodů spojených pružinami*. Nevýhodou tohoto přístupu je, že *nelze dosáhnou dokonalé tuhosti těles*. To je způsobeno omezenou možností nastavení *tuhosti pružin.* Tělesa pak při simulaci vypadají jakoby byla z gumy [8].

Mezi dvěma částicemi je pružina, která má klidovou délku l_0. Což je délka pružiny, při níž *nepůsobí na systém žádná vnější síla.* Pokud se pružina smrští nebo natáhne, působí na obě částice na jejím konci síla (viz výše: pohyb tělesa na pružině). Velikost této síly se určí jako: $\boldsymbol{F} = -k.(x - l_0)$, kde *k* je tuhost pružiny, a *x* vzdálenost bodů po prodloužení pružiny [13], viz obrázek 10. Více o pohybu na pružině viz výše. Po skončení silového působení by pružiny systému kmitaly donekonečna. Proto se zavádí tzv. *tlumení*. Může být například způsobeno vnitřním tření pružin. Toto tlumení pak samozřejmě způsobí ustálení pružin opět do klidového stavu [12].

Výsledná síla ***F*** je tedy zmenšena o velikost *tlumící síly* a je aplikována na obě tělesa, s tím rozdílem, že s vzájemně opačným směrem (viz třetí Newtonův zákon akce a reakce).



Obrázek 10: působení pružin ve spring-mass systému

4. Implementace fyzikálního simulátoru v Javě

4.1 Úvod

Cílem praktické části mé bakalářské práce byla implementace jednoduchého fyzikálního simulátoru ve 2D prostředí, ve kterém budou brány v úvahu jednak vlastnosti objektů (hmotnost, poloměr, pružnost,…) a další síly (gravitace, tření, …) a interakce (detekce a řešení kolizí,…) působící v reálném prostředí.

Mnou realizovaný fyzikální simulátor je desktopová aplikace implementovaná v jazyce Java. Pro vykreslování dvourozměrné grafiky jsem použil API Java2D. Při implementaci simulátoru jsem se snažil dosáhnout co největší přesnosti (zvláště při výpočtech hodnost z fyz. vzorců). Z tohoto důvodu je aplikace velice výkonnostně náročná. Všechna tuhá tělesa, která jsou simulována, jsou koule, protože jsou to nejjednodušší tělesa pro simulaci. Implementace simulace těles jiných tvarů by byla mnohem náročnější.

Aplikaci lze rozdělit do tří oddělených simulací:

- *Simulace tělesa na pružině* – simulace chování tělesa při různé hmotnosti a tuhosti pružiny. Více viz níže.
- *Simulace těles pohybujících se v tíhovém poli* – jedná se o rigid-body simulaci. Zahrnuje chování těles v tíhovém poli, odrazy od podložky, vzájemné kolize mezi tělesy, působní tření… V samotné aplikaci je tato část ještě rozdělena do tří dalších částí: *fyz. simulace kolize tělesa a povrchu, fyz. simul. valení tělesa po povrchu* a *fyz. simul. kolizí mezi tělesy*. Více viz níže.
- *Simulace lana* – Simulace založená na spring-mass systému. V aplikaci je rozdělena do dvou dalších části: *fyz.simulace lana* a *fyz. simulace tělesa zavěšeného na laně.* Více viz níže.

V prvních dvou případech je *simulace 10x zpomalena*, což umožnilo provádět přesnější výpočty. V posledním případě je rychlost simulace volitelná.

4.2 Čas, volba časového kroku

Jak už bylo v kapitole o analýze problému fyzikálních simulací zmíněno, nelze dosáhnout zcela přesné simulace, protože *počítače nejsou schopny* v reálném čase efektivně analyticky řešit diferenciální rovnice pohybu. Proto simulace probíhá po časových krocích *dt* a tento časový krok je pak aproximován nějakou numerickou metodou.

Ve své práci jsem využil nejjednodušší numerickou metodu – *Eulerovu metodu*. Tato metoda se využívá také v počítačových hrách. Velmi jednoduše se podle ní vypočítává rychlost a poloha pro další časový úsek s ohledem na působící sílu a čas. Tyto vzorce jsou uvedeny výše v kapitole o Eulerově metodě.

Při volbě *dt* je jasné, že čím *menší bude, tím realističtější bude simulace*[4]. Při volbě velkého *dt* se nebude simulace chovat realisticky a bude docházet k nestabilitě simulace. V automobilovém simulátoru je vhodné volit *dt* od 1 do 5ms. V arkádovém simulátoru pak od 10 do 200ms [1].

V mém simulátoru *v částech simulace tělesa na pružině* a *simulace těles v tíhovém poli* jsem využil toho, že simulace je 10x zpomalená, *což umožňuje volit velmi malé dt.* Aplikace tam využívá časový krok *dt=0,2ms. V části simulace lana je dt proměnné* v závislosti na zvolené rychlosti simulace *od 0,2* po *2ms.*

Jelikož je simulace zpomalena a různé simulované akce probíhají různou dobu v závislosti na náročnosti výpočtu (let tělesa vzduchem je mnohem méně výkonnostně náročný než výpočet tření valícího se tělesa), je v aplikaci uváděn *skutečný čas simulace*. Ten se stanovuje jako součet všech nasimulovaných časových kroků *dt*. $Celkový\ čas = i * dt$, kde *i* je počet cyklů simulace.

4.3 Třídy realizující grafickou část aplikace

Pro realizaci grafické části aplikace jsou užity především prvky knihovny *Swing*. Částečně je i využito knihovny *AWT*. Pro realizaci vykreslování dvourozměrné grafiky je použito API Java2D, které je jednoduché a dostačující pro tuto práci.

Třída PhySimul2D

Třída PhySimul2D je hlavní třídou aplikace. Vytváří hlavní rám aplikace. Je v ní definováno hlavní rozvržení aplikace, boční panel (`Panel`) pro výběr a zadávání parametrů simulace, a skutečná plocha, ve které probíhá simulace (instance třídy `Area`) a menu aplikace. Jsou zde nastaveny rozměry plochy simulace v metrech a počet pixelů na jeden metr.

Tato třída také zprostředkovává komunikaci mezi simulací a bočním panelem s parametry simulace. Zpracovává události instance třídy `Panel` a přeposílá je konkrétní simulaci.

Dědí ze třídy `JFrame`.

Hlavní metody:

- `menuInit()` - definování a vytvoření menu.
- `AreaInit()` - definování plochy a bočního panelu, zpracovávání událostí instance třídy `Panel`.
- `Init()` – definování rozměrů rámu, viditelnosti, titulku,…

Třída Panel

Třída Panel dědí ze třídy `JPanel`. Reprezentuje levý panel pro výběr simulace. Umožňuje zadávání parametrů simulace, zobrazovaní některých dynamických i statických parametrů dané simulace, spouštění a zastavování simulace.

Její události jsou zpracovávány ve třídě `PhySimul2D`.

[4] Nelze však volit zcela malé z několika důvodů. Viz kapitola fyzika a počítače.

Třída Area

Třída Area dědí ze třídy JPanel a implementuje rozhraní Runnable. Definuje plochu pro simulaci. Zajišťuje vykreslování plochy, simulovaných těles, času, atd. Definuje instanci třídy `Simulation`, která zajišťuje samotnou simulaci. Uchovává informace o uplynulém čase, časovém kroku, a velikosti zpomalení simulace.

Zajišťuje cyklické přepočítávání časového kroku *dt* a volání metody `operate(double dt)` instance třídy `Simulation`. Tato metoda uskutečňuje simulaci za časový krok dt.

Třída Area je společným předkem pro třídy AreaForMWS, AreaForMUG a AreaForRS, které jsou již konkretizovány pro jeden ze tří druhů simulace.

Hlavní metody:

- `Area(…)` – zajišťuje definování plochy.
- `start()` – metoda spouštějící provádění simulace.
- `run()` – metoda provádějící cyklické přepočítávání *dt*, volání metody `operate(dt)` instance třídy `Simulation`, volání metody `repaint()` a odesílání hodnost simulace k vypsání v GUI.
- `paint(Graphics g)` - zajišťuje překreslování plochy, atd. V potomcích je tato metoda překryta.

Třída AreaForMWS

Dědí ze třídy Area. Je specializována pro simulaci tělesa zavěšeného na pružině (MassConnectedWithSpring). Má definovány parametry určené pro simulaci tělesa na pružině (`springConstant, gravitation,…`). V konstruktoru je inicializována proměnná `simulation` na instanci třídy `MassConnectedWithSpring`.

Hlavní metody:

- `paintArea(Graphics g)` – metoda pro vykreslení plochy (tj. pozadí, okraje).
- `paintGrid(Graphics g)` – metoda pro vykreslování pomocné mřížky.
- `paintTime(Graphics g)`- metoda pro vykreslování uplynulého času simulace.
- `paintSpringAndMass(Graphics g)` – metoda pro vykreslování pružiny a tělesa.
- `paintVelocity(Graphics g)` - vykreslování šipky ukazující směr a velikost rychlosti.

Všechny tyto metody jsou volány metodou `paint(Graphics g)`.

Třída AreaForMUG

Dědí ze třídy Area. Je specializována pro simulaci těles pohybujících se v tíhovém poli (MotionUnderGravitation). Má definovány parametry určené pro simulaci těles v tíhovém poli, seznam simulovaných těles, atd. V konstruktoru je inicializována proměnná `simulation` na instanci třídy `MotionUnderGravitation` a je definován povrch (instance třídy `Ground`).

Hlavní metody:

- `paintArea(Graphics g)`, `paintGrid(Graphics g)`, `paintTime(Graphics g)`, `paintVelocity(Graphics g)` – obdoba metod ve třídě AreaForMWS.

- `paintGroud(Graphics g)` – metoda pro vykreslení tvaru povrchu pomocí úseček.
- `paintFreeBodies(Graphics g)` – metoda pro vykreslení simulovaných těles.
- `paintFriction(Graphics g)-` metoda pro vykreslení šipky ukazující směr a velikost třecí síly působící na těleso.

Třída AreaForRS

Dědí ze třídy Area. Je specializována pro simulaci lana (RopeSimulation). Má definovány parametry pro simulaci lana a tělesa zavěšeného na laně (`groundHeight`, `numOfMassesRope`, `mOfHungMass`, …). V konstruktoru je inicializována proměnná `simulation` na instanci třídy `RopeSimulation`, je nastavena rychlost simulace, atd.

Hlavní metody:

- `paintArea(Graphics g), paintGrid(Graphics g), paintTime(Graphics g), paintVelocity(Graphics g)` – obdoba metod ve třídě AreaForMWS.
- `paintGroud(Graphics g)` – metoda pro vykreslení povrchu v určité výšce.
- `paintRope(Graphics g)` – metoda pro vykreslení lana a zavěšeného tělesa.

4.4 Třídy matematické knihovny aplikace

Pro implementaci fyzikálního simulátoru je nezbytné mít matematické knihovny. Část matematické knihovny, která je v simulátoru použita, je ze standardních knihoven jazyka Java (např. třída `Math`). Jiné jsem naimplementoval sám.

Třída Vector2D

Reprezentuje dvourozměrný vektor v prostoru. V aplikaci představuje body, pozice, vektory, rychlost, sílu,… ve 2D prostoru. Obsahuje souřadnice x, y typu double. Zavádí metody pro práci s vektory.

Hlavní metody:

- `add(Vector2D v), sub(Vector2D v)` – přičítá (odečítá) od aktuálního vektoru vektor *v*.
- `addR(Vector2D v), subR(Vector2D v)-` sčítá (odečítá) dva vektory a vrací výsledný vektor.
- `mul(double d), div(double d)` – násobí (dělí) vektor hodnotou a vrací výsledný vektor.
- `scalarProduct(Vector2D v)` – metoda vrací skalární součin dvou vektorů.
- `getAngleAlga(Vector2D v)` – metoda vrací úhel, který svírají dva vektory ve stupních.
- `AngleAlfa()` – metoda vrací úhel, který svírá vektor s y-ovou osou ve stupních.
- `turnAboutAngle(double alfa)` – pootočení vektoru o úhel alfa.
- další metody pro práci s vektory (otočení vektoru, Eulerova vzdálenost, …).

Třída Mathh

Třída Mathh je třída, která obsahuje pouze dvě statické metody pro řešení rovnic.

Metody:

- `quadraticeQuation(double a, double b, double c)` – statická metoda řešící výpočet kořenů kvadratické rovnice. Kořeny vrací ve formě vektoru `Vector2D`. V případě nenalezení se vrací vektor s hodnotami `Double.MAX_VALUE`.
- `linearQuation(double a1, double b1, double c1,double a2, double b2, double c2)` – Statická metoda řešící výpočet soustavy dvou rovnic od dvou neznámých, kde `a1, b1, c1` jsou parametry první rovnice a `a2, b2, c2` parametry druhé rovnice. Hodnoty *x* a *y* vrací ve formě vektoru `Vector2D`. V případě nenalezení vrací `null`.

Třída Line

Instance třídy Line reprezentuje přímku ve dvourozměrném prostoru. Tato přímka je definována jedním bodem, směrnicovým vektorem, normálovým vektorem a parametry obecné rovnice přímky *a*, *b*, *c*.

Hlavní metody:

- `Line(double a, double b, double c)` - konstruktor, který vytváří instanci třídy na základě parametrů obecné rovnice přímky *a*, *b*, *c*.
- `Line(double a, double b, Vector2D point)` - konstruktor, který vytváří instanci třídy na základě parametrů obecné rovnice přímky *a*, *b* a bodu ležícího na přímce.
- `Line(Vector2D point1, Vector2D point2)` - konstruktor, který vytváří instanci třídy na základě dvou bodů, přes které přímka prochází.
- `IntersectionLines(Line l)` – metoda, která vrací průsečík dvou přímek ve formě vektoru `Vector2D`.
- `getParamTForPoint(Vector2D point)` – metoda vrací parametrickou vzdálenost bodu (`point`) od bodu definovaného v přímce.
- `getPointOnLine(double t)` – metoda vrací bod na přímce ve formě vektoru `Vector2D`, který je od bodu definovaného v přímce vzdálen parametrickou vzdálenost `t`.
- `pointOnLine(double x)` – metoda, která vrací bod na přímce ve formě vektoru `Vector2D`, když je známa jeho x-ová souřadnice.

Třída LineSegment

Instance Třídy LineSegment reprezentuje úsečku ve 2D. Dědí ze třídy Line. Je definována dvěma koncovými body.

Hlavní metody:

- `IntersectionLineSegments(Line l)` – metoda určuje průsečík dvou úseček a vrací tento průsečík jako instanci třídy `Vector2D`. V případě nenalezení průsečíku vrací `null`.
- `IntersectionLineSegmentAndLine(Line l)` - metoda určuje průsečík úsečky a přímky a vrací tento průsečík jako instanci třídy `Vector2D`. V případě nenalezení průsečíku vrací `null`.

4.5 Třídy simulační knihovny aplikace

Třídy simulační knihovny tvoří samotné jádro aplikace. Starají se o reprezentaci simulovaných objektů a realizaci samotné simulace. Ke svému chodu využívají matematické knihovny.

Třída Mass

Tato třída reprezentuje simulované těleso ve 2D. V případě tohoto fyz. simulátoru je to koule nebo bezrozměrná částice. Jsou zde definovány všechny vlastnosti simulovaného tělesa, tj. hmotnost, poloměr, rychlost, síla působící na těleso a jeho pozice v prostoru.

Tato třída realizuje nulování a aplikování sil na těleso a *především simulaci tohoto tělesa v časovém kroku dt.*

Hlavní metody:

- `init()` – vynulování působící síly na těleso.
- `applyForce(Vector2D force)` – aplikace vnější síly na těleso.
- `simulace(double dt)` – metoda zajišťující nasimulování tělesa za *dt*. Tj. na základě výpočtů podle Eulerovy metody, stanovení nové rychlosti a pozice tělesa.

Třída Simulation

Při fyzikální simulaci se v každém časovém kroku opakuje totéž. Síly musí být nejdříve vynulovány a pak znovu vypočítány a nakonec se provede, v závislosti na těchto silách, přepočtení polohy a rychlosti všech těles, které v simulaci účinkují. Úkolem třídy Simulation je tedy se starat o celý běh simulace. Tato třída si udržuje informace o objektech simulace a jejich počtu a umožňuje jejich vytváření.

Třída Simulation je pouze obecným předkem pro třídy MassConnectedWithSpring, MotionUnderGravitation a RopeSimulation.

Hlavní metody:

- `init()` – tato metoda volá na všechny objekty simulace jejich metodu `init()` pro vynulování síly.
- `simulate(double dt)` – metoda řešící všechny výpočty simulace za *dt*. Překryta v potomcích.
- `operate(double dt)` – kompletní simulační metoda, volá metody `init()` a `simulate(dt)`.

Třídy MassConnectedWithSpring, MotionUnderGravitation a RopeSimulation, Spring a Ground, které patří také do simulační knihovny, budou rozebrány v následujících kapitolách.

4.6 Realizace fyz. simulace tělesa zavěšeného na pružině

První ze tří částí aplikace je fyzikální *simulace pohybu tělesa na pružině*. Při této simulaci je bráno v úvahu působení tíhové síly na pružinu s tělesem, hmotnost tělesa, délka pružiny bez zatížení a její

požadované prodloužení při zavěšení tělesa a nakonec také tuhost pružiny. Je zde zanedbáno působení třecích sil v pružině a také působení odporu vzduchu na těleso. To znamená, že těleso se nikdy na pružině nezastaví.

Veškeré fyzikální zákony a interakce, které zde bylo potřeba implementovat, jsou již zmíněny v předchozích kapitolách.

O realizace této simulace se stará třída MassConnectedWithSpring.

MassConnectedWithSpring

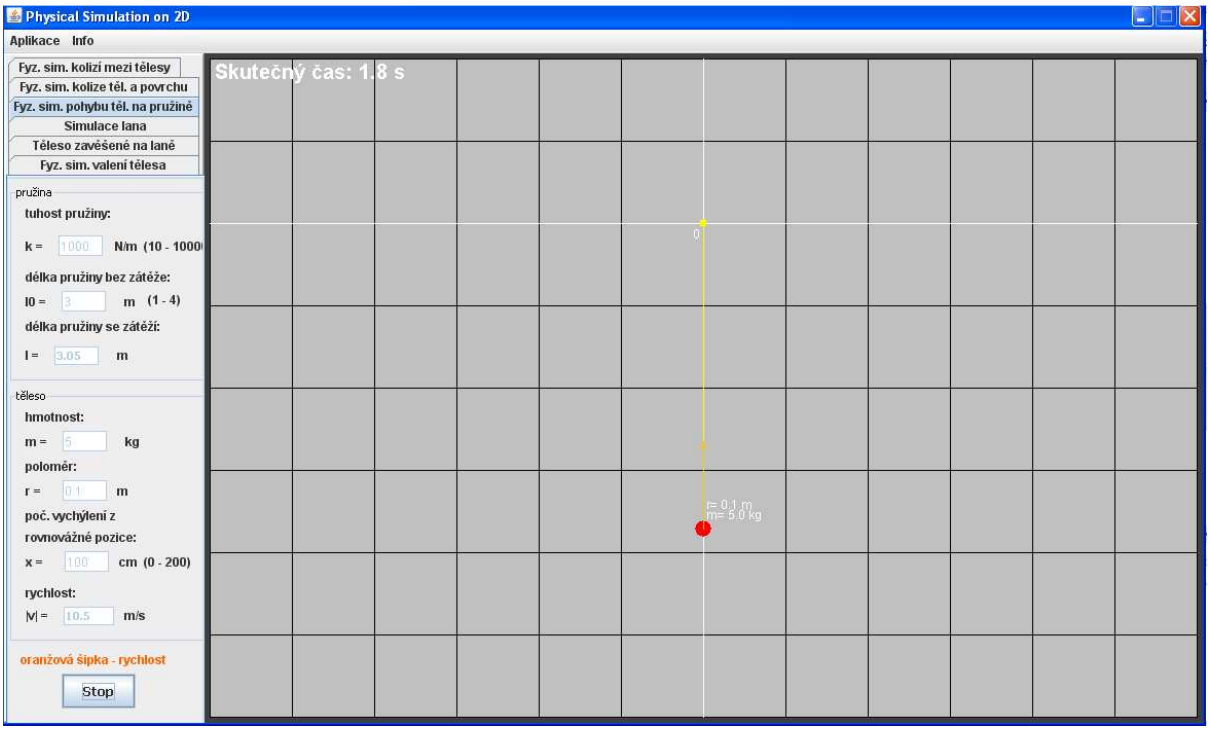
Třída dědící ze třídy Simulation. Je specializována pro realizaci a provádění výpočtů pro fyz. simulaci tělesa zavěšeného na pružině.

V konstruktoru probíhá vytváření tělesa (instance třídy Mass), stanovování nové délky pružiny a vypočítávání *rovnovážného bodu* tohoto pohybu na pružině.

Hlavní metody:

- `solve()` – vypočítávání působení sil v daném časovém okamžiku a jejich nastavování tělesu.
- `simulate ()` – volá metodu `solve()`a spouští simulaci na simulovaném tělese.

Ukázka této fyzikální simulace z aplikace je na obrázku 11.



Obrázek 11: ukázka fyzikální simulace tělesa zavěšeného na pružině

4.7 Realizace fyz. simulace těles pohybujících se v tíhovém poli

Druhou částí aplikace je fyzikální *simulace těles pohybujících se v tíhovém poli*. V aplikaci se tato fyz. simulace pro názornost dělí ještě do dalších tří částí a to *fyzikální simulace valení tělesa po šikmé ploše, fyzikální simulace kolizí tělesa a povrchu* a *fyzikální simulace kolize mezi tělesy.*

V této simulaci se jedná o simulaci *tuhých těles tzv. rigid-body simulace* (viz výše). Všechna simulovaná tělesa jsou koule, protože koule je nejjednodušší těleso pro simulaci. Všechny výpočty sil, odrazů a jiných jsou zde počítány co nejkorektněji podle fyzikálních zákonů. Je zde bráno v úvahu hmotnost a poloměr tělesa, působení tíhové síly, odpor vzduchu, vlastnosti nakloněné roviny, působení třecích sil při valení tělesa, neideálnost povrchu (součinitel restituce). Naopak je zde zanedbáno neideálnosti samotných těles, tzv. že např. srážka dvou těles je simulována jako srážka ideální.

Realizace této fyz. simulace byla nejnáročnější vzhledem k nutnosti řešit kolize jak se statickým tělesem (povrchem) tak mezi tělesy navzájem. Bylo zde potřeba řešit velké množství problému jak z hlediska fyziky tak matematiky. Nastínil bych zde některé z těchto problému a jejich řešení.

Realizace povrchu, třída Ground

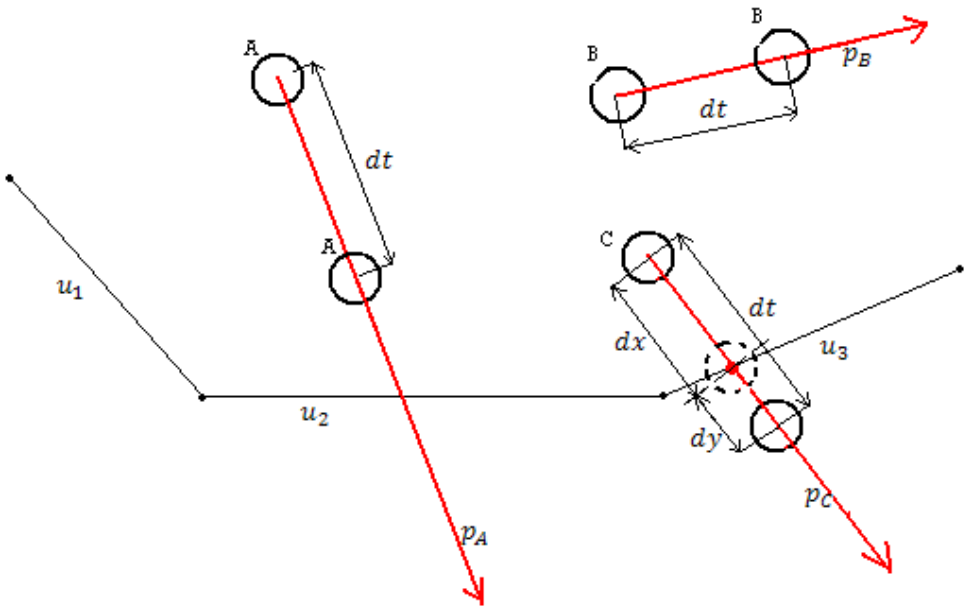
Povrch je v simulaci považován za statické těleso. Je realizován třídou `Ground`. V této třídě je definován jako sada úseček (třída LineSegment), které zde jsou uspořádány podle jejich umístění v prostoru podle velikosti x-ové souřadnice jejich bodů (tj. zleva doprava).

Kolize tělesa a povrchu

Zde bych chtěl naznačit problém kolizí tělesa a povrchu a jeho řešení, které jsem použil ve své práci.

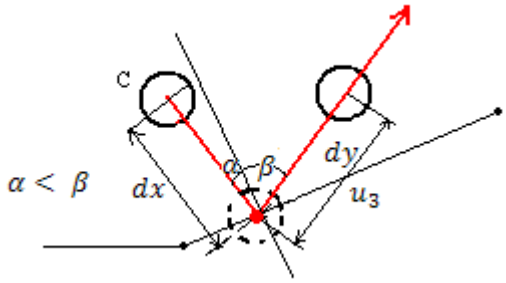
Problém kolizí se obecně skládá ze dvou částí: *detekování kolize* a *reakce těles na náraz.*

Při detekcí kolizí *je nutné* při každém časovém kroku *testovat* pro každé těleso, jestli bude v tomto časovém kroku kolidovat. Toto nelze určit jinak než s pomocí analytické geometrie s využitím přímek, polopřímek a úseček. Vysvětlení je na obrázku 12. Těleso B směřuje ve směru polopřímky p_B, *a tedy směřuje zcela mimo definovaný povrch*, který je definován úsečkami u_1, u_2, u_3. Kolize není potřeba řešit. Těleso A směřuje ve směru polopřímky p_A, *směřuje směrem k úsečce* u_2, *je potřeba otestovat*, zda v tomto časovém kroku dt nedojde ke kolizi. Po otestování je zjištěno, že ne a kolizi není potřeba řešit. V případě tělesa C, jeho polopřímka protíná úsečku u_3, *a dochází zde ke kolizi v časovém kroku dt v čase dx.* V *dy* se těleso už pohybuje pod zemí.



Obrázek 12: detekce kolize tělesa a povrchu

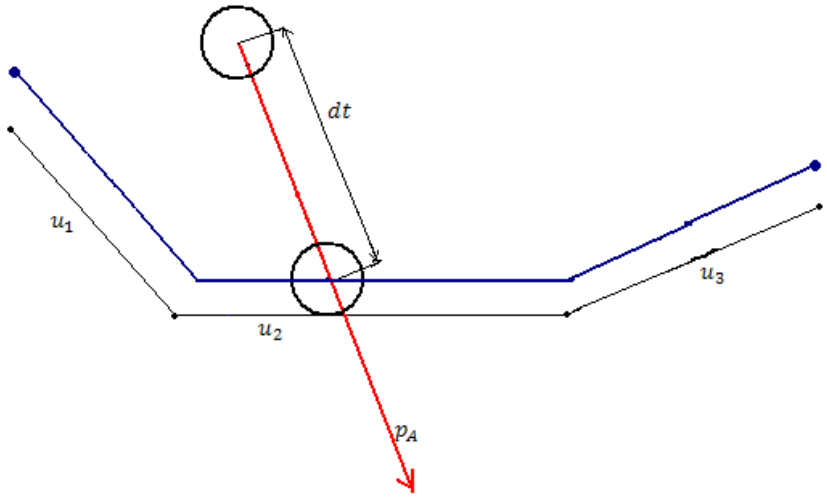
Při řešení kolize je potřeba určit přesné hodnoty *dx* a *dy*. To lze určit z rovnic nerovnoměrného pohybu (viz kapitola 2.2). Dále je provedeno nasimulování tělesa za čas *dx* do kolizního bodu. Nyní je potřeba vypočítat odraz tělesa a jeho novou polohu s ohledem na sklon šikmé plochy (otáčení souřadnic), součinitel restituce (řešení problému výpočtu úhlu odrazu) v času *dy*. Viz obrázek 13.



Obrázek 13: řešení kolize a odraz

Kolizní obálka

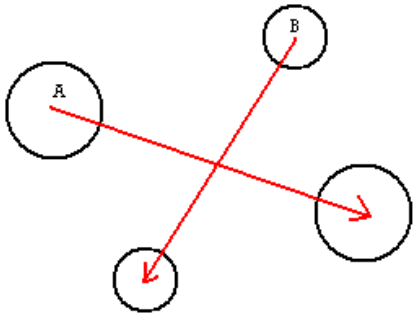
Pro vyřešení kolize zbývá ještě jeden problém a to, že těleso je svou polovinou, v době odrazu, stále pod zemí. Je potřeba proto posunout povrch o poloměr tělesa směrem k tomuto tělesu. Viz obrázek 14.



Obrázek 14: kolizní obálka

Kolize mezi dvěma tělesy

Koule je v prostoru reprezentována středem a poloměrem. Určení zda dvě koule do sebe narazily je jednoduché. Součet poloměrů musí být větší nebo roven než součtu vzdáleností jejich středů. *Problém nastává, když se tyto dvě koule pohybují*. Na obrázku 15. je příklad, kdy se dvě koule přesunou z jednoho místa do druhého za určitý časový úsek. Jejich dráhy se protínají, ale to není ještě dostatečný důvod k tvrzení, že do sebe opravdu narazí. Mohou se například pohybovat rozdílnou rychlostí [5].



Obrázek 15: srážka těles

Při detekování srážky dvou koulí nelze použít stejné metody jako při srážce koule a povrchu. Metod jak odhadnout (co nejpřesněji určit) čas a místo srážky je několik. Já jsem ve své práci nezvolil zrovna nejideálnější řešení, ale v dané situaci nebyla jiná možnost. Pro každé těleso po nasimulování jeho pozice za *dt* je testováno, zda není v kolizi s nějakým jiným tělesem. Pokud ano je řešena tato kolize (popis řešení kolize zde nebudu popisovat, vychází z fyz. vzorců popsaných v kapitole 2.6). Nevýhodou tohoto postupu je, že jedno z kolidujících těles je považováno za nepohybující se (místo kolize pak není přesně určeno). Druhou podstatnou nevýhodou je, že toto druhé nepohybující se těleso *může kolidovat o jeden časový krok dřív.* A nakonec taky, že tělesa se při kolizi částečně překrývají. Jelikož je v simulátoru použit velmi malý časový krok (*dt*=0,2ms), tyto na první pohled velké nevýhody, nejsou vůbec tak významné, aby nějak zásadně ovlivnili simulaci.

Valení tělesa po povrchu

Problém s valením tělesa po povrchu je především ten, že *je potřeba určit, kdy se jedná ještě o odraz a kdy už o valení tělesa*. Rozeznání těchto dvou stavů je potřeba především pro realističtější chování simulace a také z hlediska aplikace třecí síly (ta působí jen u valení).

V práci je použito k rozeznání odrazu a valení tento způsob: pomocí matematických vzorců je určena *velikost rychlosti tělesa ve směru plochy*. Tato rychlost je porovnána s *velikostí rychlosti tělesa*. Pokud jsou si tyto rychlosti velmi blízké (tisíciny), pak *se jedná o valení* a jsou aplikovány příslušné síly v závislosti na naklonění roviny, atd., jak o nich bylo psáno v kapitole 2.5.

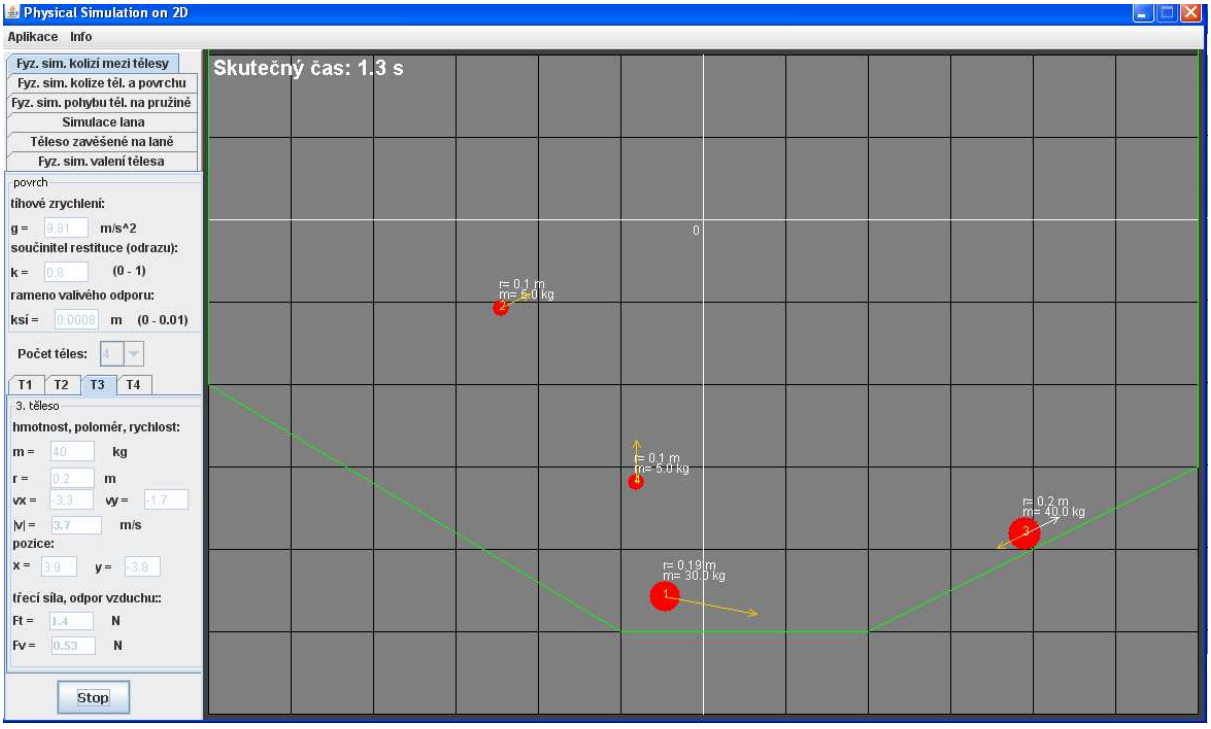
Třída MotionUnderGravitation

Třída dědící ze třídy Simulation. Je specializována pro simulaci těles pohybujících se v tíhovém poli.

Hlavní metoda:

- `simulate ()` – V této metodě se cyklicky provádí pro každé těleso aplikace sil, detekování kolizí, řešení kolizí, provádění veškerých simulačních výpočtů (velikost, směr třecí síly, tíhové síly,…), výpočtů chování tělesa po srážce, volání metody `simulace(dt)` u každého tělesa, atd.

Ukázka této fyzikální simulace z aplikace je na obrázku 16.



Obrázek 16: fyzikální simulace těles pohybujících se v tíhovém poli

4.8 Realizace fyz. simulace lana

Poslední části aplikace je *fyzikální simulace lana*. V aplikaci se tato simulace dělí do dvou částí: *simulace lana* (i s odrazem od země) a *simulace tělesa zavěšeného na laně*.

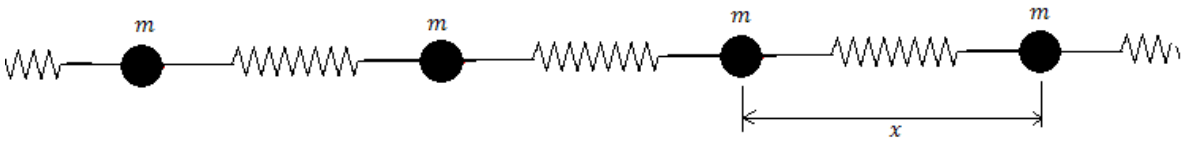
Tato simulace byla naimplementována podle příkladu uvedeného v [13].

Jedná se zde simulaci s použitím tzv. *spring-mass systémů (viz výše)*. Simulovaný objekt - lano *je pokládáno z velkého množství částic.*

V této simulaci je bráno v úvahu působení tíhové síly, hmotnost lana, působení odporu vzduchu na lano, působení třecí síly v pružinách, působení smykového tření, řešení odrazu od povrchu. Na rozdíl od předchozí simulace, kde byly všechny síly a interakce počítány „zcela“ přesně, zde není brán zřetel na úplnou fyzikální přesnost. To ovšem ale na druhou stranu umožňuje mnohem rychlejší a plynulejší simulaci.

Realizace lana

Lano *je složeno z velkého množství částic* (instance třídy Mass), které jsou uspořádány v řadě za sebou. Počáteční vzdálenost každých dvou sousedních částic je stejná (např. 5cm). *Částice jsou hmotné* (součet hmotnosti všech částic je rovna hmotnosti lana), ale bezrozměrné (na rozdíl od předchozích simulací). *Mezi každými dvěma částice umístěna pružina* (instance třídy Spring). Díky této pružině se udržuje soudržnost lana a vzniká zde vzájemné předávání síly mezi částicemi. Délka pružiny v klidovém stavu je rovna počáteční vzdálenosti dvou částic. Viz obrázek 17. Při rozpohybování lana začne docházet, díky těmto pružinám, k silovému působení mezi částicemi, viz kapitola 3.3 o spring-mass systémech.



Obrázek 17: realizace lana

Třída Spring

Tato třída reprezentuje pružinu. Popisuje dvě částice a silové působení na každou z nich. Jsou v ní definovány tyto dvě částice, tuhost pružiny a délka pružiny (vzdálenost částic) při které nepůsobí žádné síly.

Hlavní metoda:

- `solve ()` – metoda se stará o výpočet sil působících na těleso udělovaných pružinou, s přihlédnutím na tření v pružině a jejich aplikaci na obě částice.

Třída RopeSimulation

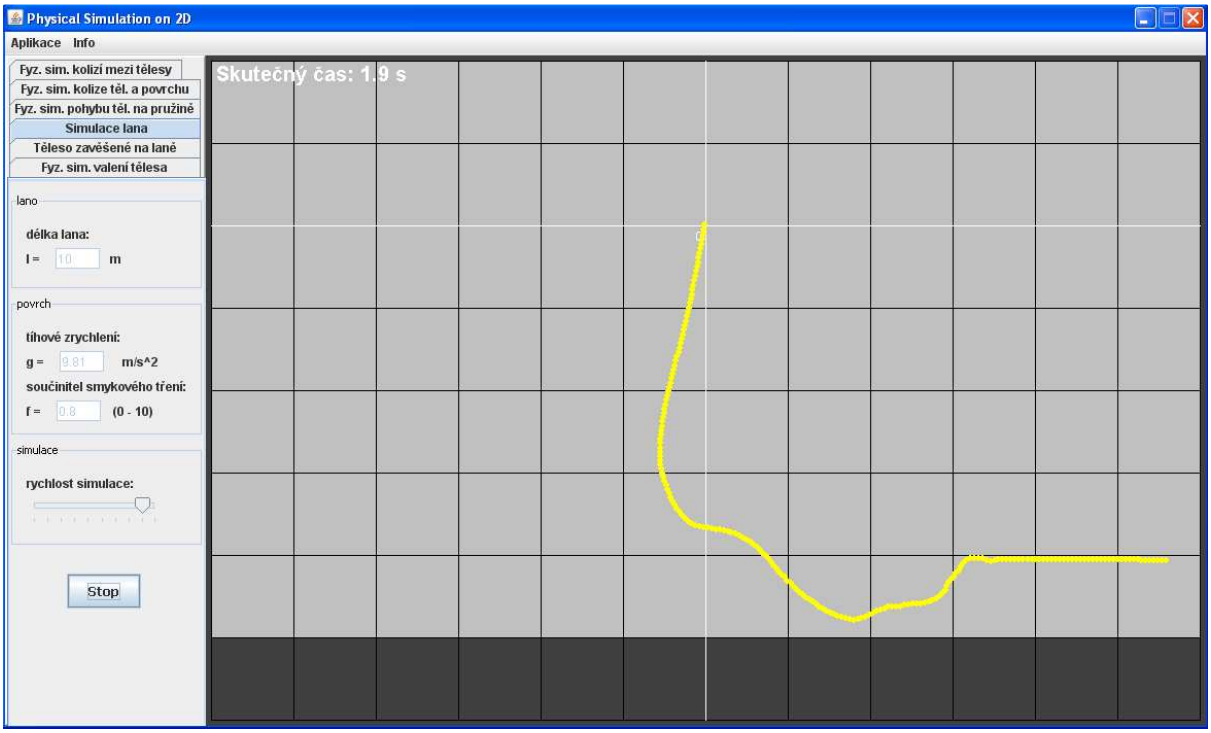
Třída dědící ze třídy Simulation. Je specializována pro realizaci a provádění výpočtu pro *simulaci lana* a *simulaci tělesa zavěšeného na laně.*

V konstruktoru probíhá nastavování parametrů simulace, vytváření lana, tj. vytváření potřebného počtu částic a jejich umísťování v prostoru a vytváření pružin mezi nimi. V případě *simulace tělesa zavěšeného na laně* také vytváření tohoto tělesa.

Hlavní metody:

- `solve()` – metoda pro výpočet působících sil na každé těleso vzniklých působením pružin, tíhovou silou, odporem vzduchu, třecí silou, atd. a jejich aplikace pro každé těleso.
- `simulate ()` – volá metodu `solve()`a spouští simulaci pro každou částici lana.

Ukázka této fyzikální simulace z aplikace je na obrázku 18.



Obrázek 18: fyzikální simulace lana

5. Testy náročnosti na výkon počítače a možnosti optimalizací

Testování bylo prováděno na dvou simulacích aplikace a to při *fyzikální simulaci pohybu těles v tíhovém poli* a při *simulaci lana*. Na první simulaci byly provedeny dva testy a druhé jeden test.

Testy byly prováděny na počítačové sestavě s touto hardwarovou konfigurací: procesor Intel® Core2 Duo™ T5270 1,4GHz, operační paměť o velikosti 2GB a grafickou kartou ATI Mobility Radeon™ HD 2400 XT s pamětí 256 MB. Operační systém počítače byl Microsoft Windows XP Service Pack 3. Během testů na počítači nebylo nic jiného spuštěno.

5.1 Testování simulace těles pohybujících se v tíhovém poli

Na fyzikální simulaci pohybu těles v tíhovém poli byly provedeny dva testy. Oba byly zaměřeny na zjištění skutečné doby trvání simulace při simulování určitého časového úseku a doby trvání simulace jednoho časového úseku *dt* při *různém počtu simulovaných těles*. Pro každý parametr byly provedeny tři měření a z nichž pak určena výsledná hodnota.

Test 1

V tomto testu byla měřena doba trvání nasimulování 30 sekund fyzikální simulace pro 1, 2, 5, 10 simulovaných těles s časovým krokem *dt* =0,2 ms. V tomto testu docházelo ke kolizím těles s povrchem a mezi sebou navzájem. *Nedocházelo zde k valení těles po povrchu!*

Fyzikální simulace je zde 10x zpomalená.

Výsledky testu lze vidět v tabulce 1.

počet těles	doba trvání 30 sekund simulace (s)	doba trvání výpočtu jednoho dt (ms)
1	296,2	1,97
2	302,1	2,01
5	386,3	2,58
10	704,6	4,7

Tabulka 1: výsledky 1. testu

Z výsledků lze vidět, že při malém počtu těles probíhá simulace, tak jak bychom si přáli. Doba výpočtu jednoho *dt* také odpovídá požadovaným hodnotám. *Při zvyšování počtu těles začíná časová náročnost simulace exponenciálně růst* (viz tabulka 1). To je způsobené jednak větším množstvím těles, které je potřeba simulovat za čas *dt* a *také zvětšujícím se počtem vzájemných kolizí mezi tělesy*.

Test 2

V tomto testu byla také měřena doba trvání nasimulování 30 sekund fyzikální simulace pro 1, 2, 5, 10 simulovaných těles s časovým krokem *dt* = 0,2 ms. V tomto testu docházelo ke kolizím těles s povrchem a mezi sebou navzájem. *A docházelo zde minimálně od poloviny testovaného času k valení některých simulovaných těles po povrchu!*

Fyzikální simulace je zde opět 10x zpomalená.

Výsledky testu lze vidět v tabulce 2.

počet těles	doba trvání 30 sekund simulace (s)	doba trvání výpočtu jednoho dt (ms)
1	296	1,97
2	301,7	2,01
5	408,2	2,72
10	752,2	5,01

Tabulka 2: výsledky 2. testu

Tento test dává obdobné výsledky jako test první. *Při zvyšování počtu simulovaných těles začíná časová složitost velmi rychle (exponenciálně) růst.* Oproti prvnímu testu zde lze vidět vyšší naměřené hodnoty. Zvláště je to patrné při větším počtu těles. *Z toho lze usuzovat, že výpočet valení tělesa zvyšuje náročnost simulace.* U malého počtu těles se ovšem tato zátěž neprojeví.

5.2 Testování simulace lana

V této simulaci byl proveden jeden test. Byl zaměřen na *zjištění skutečné doby trvání simulace pro různé délky lana*. Simulovaný čas byl 20 sekund s časovým krokem simulace *dt*=2 ms. Test probíhal pro délky 1, 3, 6 a 10 metrů. Pro každé parametry byly provedeny tři měření a z nichž pak určena výsledná hodnota. V tabulce 3 je také možno vidět kolik částic bylo potřeba simulovat pro danou délku lana.

Simulace probíhala v reálném čase, nebyla zpomalena.

Výsledky testu lze vidět v tabulce 3.

délka lana (m)	počet částic lana	doba trvání 20 sekund simulace (s)	doba trvání výpočtu jednoho dt (ms)
1	20	19,6	1,96
3	60	19,6	1,96
6	120	19,7	1,97
10	200	19,9	1,99

Tabulka 3: výsledky testu pro simulaci lana

Z výsledků testu lze vidět, že při různých délkách lana, se náročnost výpočtu nikterak výrazně nezvyšuje.

Z provedených testů je vidět, že druh simulace použitý u simulace těles v tíhovém poli je výkonnostně náročnější než ten, který je použitý u fyzikální simulace lana.

5.3 Možnosti optimalizací

V této kapitole budou stručně naznačeny možností optimalizací vytvořeného fyzikálního simulátoru při zachování stejné přesnosti výpočtů. Možné optimalizace budou uvažovány z hlediska zvýšení rychlosti běhu simulace.

Možné optimalizace:

- *neprovádění výpočtu simulace u těles, která již nenávratně vyletěla ze zobrazované plochy.*
- *neprovádění výpočtu simulace u těles, která nehybně leží na vodorovné podložce* – u těles, která jsou v klidovém stavu, není potřeba provádět simulaci, protože jejich stav bude i po ní stejný.
- *nedodržování zapouzdřenosti objektů* – při dodržování zapouzdřenosti je potřeba implementace get a set metod. To zvětšuje celkovou velikost aplikace a snižuje její výkon.
- *využívat méně metod* – rozdělení kódu do více metod zpomaluje aplikaci.
- *optimalizace elementárních operací např. dělení* – dělení je jednou z nejnáročnějších operací. Proto je např. dělení mocninou dvou vhodné naradit bitovým posunem.
- *implementace simulátoru v jiném jazyce než v Javě* – jazyk Java není primárně navržen tak, aby poskytoval co nejvyšší výkon.

6. Závěr

Cílem této práce byla implementace jednoduchého fyzikálního simulátoru ve 2D prostředí, kde budou brány v úvahu jednak vlastnosti objektů tak i další síly a interakce, které působí v reálném světě. Během tvorby tohoto simulátoru jsem se seznamoval s fyzikálními, matematickými záležitostmi a problémy fyzikálních simulací, které je potřeba znát při tvorbě fyzikálního simulátoru. Toto jsem také popsal v prvních dvou kapitolách práce.

Samotný vytvořený fyzikální simulátor se pak skládá ze tří částí. První částí je fyzikální simulace tělesa kmitajícího na pružině. Druhá část je fyzikální simulace těles pohybujících se v tíhovém poli. Zde jsem použil tzv. simulace pomocí tuhých těles. Třetí částí je pak fyzikální simulace lana. Zde byla použita simulace pomocí tzv. spring-mass systémů. V simulátoru se mi podařilo realizovat působení základních fyzikálních sil a aplikování základních fyzikálních zákonů (pružnost, gravitace, tření, odpor vzduchu, …). Dále pak realizovat detekování a řešení kolizí se statickým i dynamickým tělesem, kde statické těleso je považováno za nedokonale pružné. Popis aplikace a naznačení řešení některých problémů, které bylo potřeba řešit, jsem popsal ve třetí části této práce.

Na závěr byly provedeny testy náročnosti na výkon počítače. Z těchto testů vyplynulo, že s větším počtem simulovaných těles náročnost aplikace exponenciálně roste. Úplně nakonec jsem navrhl možné optimalizace vytvořeného simulátoru.

V dalším případném rozvoji aplikace by bylo vhodné se zaměřit na optimalizování aplikace zvláště z hlediska její výpočetní náročnosti.

Seznam použité literatury

[1] TUNCA, Erkin. *Úvod do fyzikálních simulací* [online]. 2008. Dostupné z WWW: <http://nehe.ceske-hry.cz/tut_39.php>.

[2] JANDORA, Radek. *Kmitavý pohyb* [online]. Dostupné z WWW: <http://radek.jandora.sweb.cz/f10.htm#kinem>.

[3] *Učebnice matematiky a fyziky pro gymnázia* [online]. 2009. Www.ucebnice.krynicky.cz. Dostupné z WWW: <http://www.ucebnice.krynicky.cz/Fyzika/1_Mechanika/2_Dynamika/1212_Naklonena_rovina_I.pdf>.

[4] *Encyklopedie fyziky : Nakloněná rovina* [online]. 2009. Fyzika.jreichl.com. Dostupné z WWW: <http://fyzika.jreichl.com/index.php?page=98&sekce=browse>.

[5] TUNCA, Erkin. *Detekce kolizí* [online]. 2008. Dostupné z WWW: <http://nehe.ceske-hry.cz/tut_30.php>.

[6] *FyzWeb : Srážky* [online]. Dostupné z WWW: <fyzweb.cuni.cz/dilna/krouzek/index.htm>.

[7] RNDr. MIKULČÁK Jiří CSc., et al. Matematické, fyzikální a chemické TABULKY A VZORCE pro střední školy. Praha : Prometheus, 2005. 276 s.

[8] MOCNÝ, Ondřej. *Real-time fyzikální simulace pro mobilní zařízení* [online], 2009. 60 s. Bakalářská práce. Univerzita Karlova, Matematicko-fyzikální fakulta. Dostupné z WWW: <http://physics.hardwire.cz/mirror/bakalarka.pdf>.

[9] BERAN, Lukáš. *Základy fyzikální simulace v počítačových hrách* [online], 2008. Dostupný z WWW: <http://resurrection.ic.cz/soubory/GamePhysics.pdf>.

[10] Jiří Filipovič : *Simulace fyziky rigidních těles* – Záznam přednášky z Game Developers Session 2005, Dostupný z WWW: <http://www.video.muni.cz/public/gds2005/SimulaceDynamikyRigidnichTeles.wmv>.

[11] *Wikipedia : Rigid body* [online]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Rigid_body >.

[12] JELEN, Tomáš. *Modelování pohybujících se objektů pomocí částicových systémů*, 2003. 35 s. Bakalářská práce. Západočeská univerzita v Plzni.

[13] TUNCA, Erkin. *Fyzikální simulace lana* [online]. 2008. Dostupné z WWW: <http://nehe.ceske-hry.cz/tut_40.php>.

[14] *Wikipedia : Eulerova metoda* [online]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Eulerova_metoda>.